ACHIEVING REUTILIZATION OF SCHEDULING SOFTWARE THROUGH ABSTRACTION AND GENERALIZATION

George J. Wilkinson, Richard A. Monteleone,
Stuart M. Weinstein, Michael G. Mohler, David R. Zoch
Loral AeroSys
7375 Executive Place, Suite 101
Seabrook, MD 20706
(301) 805-0433

G. Michael Tong
NASA
Mail Code 522
Goddard Space Flight Center
Greenbelt, MD 20771
(301) 286-3176

ABSTRACT

Reutilization of software is a difficult goal to achieve, particularly in complex environments that require advanced software systems. The Request-Oriented Scheduling Engine (ROSE) was developed to create a reusable scheduling system for the diverse scheduling needs of the National Aeronautics and Space Administration (NASA). ROSE is a data-driven scheduler that accepts inputs such as user activities, available resources, timing constraints, and user-defined events, and then produces a conflict-free schedule. To support reutilization, ROSE is designed to be flexible, extensible, and portable. With these design features, applying ROSE to a new scheduling application does not require changing the core scheduling engine, even if the new application requires significantly larger or smaller data sets, customized scheduling algorithms, or software portability. This paper includes a ROSE scheduling system description emphasizing its general-purpose features, reutilization techniques, and tasks for which ROSE reuse provided a low-risk solution with significant cost savings and reduced software development time.

INTRODUCTION

Planning and scheduling is performed in many areas, such as spacecraft scheduling, network communications scheduling, student curriculum tracking, equipment scheduling, air traffic planning, and manufacturing. These applications share the following common attributes:

a. **Complexity**. Resource allocation for hundreds or thousands of activities may involve timing and other constraints.

b. **Limited resources**. Conflict resolution rules are needed.

c. **Competing goals**. Tradeoffs and compromises are made to reach a satisfactory solution.

d. **Automated support**. Computers can produce schedules many times faster than they can be produced manually, and can reduce errors by checking the schedules.

e. **Manual interaction**. The user must remain in control of the scheduling process, and modify schedules generated by the computer.

f. **Information display**. A highly interactive, graphical display can show complex solutions to the user, who can then modify the solutions as needed.

g. **Rescheduling**. Schedule changes may occur at any time during the process.

Often, a software development effort is undertaken to build a scheduling system for a new application. For some situations, each new implementation typically costs more than the previous ones, and is viable for that specific application only [Brouchard, 1992].

This paper describes our approach to design the ROSE software to be a reusable scheduling tool. To provide a basis for subsequent discussion, an overview of ROSE is presented first. Then, we discuss the software system characteristics, including flexibility,

extensibility, and portability, that promote and facilitate software reutilization. A detailed case study is provided for each software system characteristic. Finally, future work and a summary are presented.

OVERVIEW OF ROSE

ROSE is a generic scheduler that can be used to solve a broad range of scheduling problems, including control center applications that require interactive, real-time control over scheduling operations. Because ROSE is data driven and has built-in schedule development and rescheduling capabilities, most of the scheduling functions are already available. Custom reports and algorithms can be added to ROSE. ROSE has been demonstrated in a variety of scheduling applications (refer to Table 1).

Table 1. ROSE Applications

SCHEDULING DOMAIN	RESOURCES
Space-to-ground network communications	- Tracking and Data Relay Satellite (TDRS) antennas - Ground equipment
Data communications for the Compton Gamma Ray Observatory (GRO) satellite	- TDRS antennas
Payload scheduling scenario for the Upper Atmosphere Research Satellite (UARS)	- Science instruments - Power - Vibration
Aircraft scheduling and pilot training for the United States Air Force (USAF)	- Aircraft - Simulators - Personnel - Classrooms
Tanker aircraft scheduling and student management for the USAF	- Aircraft - Simulators - Personnel - Classrooms
Central Processing Unit (CPU) Scheduling	- CPU jobs
Global communications satellite network scheduling	- Ground station antennas - Satellite communications

ROSE is currently being integrated into NASA's Network Control Center (NCC) to provide a major capabilities upgrade with new requirements, such as flexible scheduling [Moe, et al., 1994]. The NCC schedules space-to-ground communications for NASA spacecraft, such as the Space Shuttle. As a mission-critical component of the space network, the NCC demands a highly efficient schedule. We anticipate significant cost savings will be achieved by reutilizing ROSE instead of developing a custom scheduling component or enhancing the current system to meet the new requirements.

ROSE Timeline Manager Screen

The ROSE Timeline Manager screen shown in Figure 1 is the operator's graphical interface to the schedule. Figure 1 is a black and white depiction; the actual screen is in color. The Timeline Manager screen is composed of three major sections: the menu bar, the schedule display section, and the bottom display section.

The menu bar, located near the top of the screen, contains pull-down menus that allow users to access ROSE functions, including loading, saving, and editing schedules, generating electronic or hard copy output, and calling the automated scheduler to create schedules and resolve conflicts.

The schedule display section contains graphical plots of scheduled activities, remaining resources, and other time-related scheduling information. The buttons on the left side of the graphical plots specify resources that can be selected. Each row can display the scheduled activities which use that resource. Each activity is assigned a user-defined color. Several menu functions, such as mark, annotate, cut, paste, scroll, zoom, and reschedule, are available for manipulating timelines and activities.

In addition to the activity plots, ROSE displays three other plots in this area of the screen: timegraph, step, and resource. A timegraph, displayed using white rectangles, contains one or more time windows used to characterize user events. A step plot provides more detailed information about the scheduled activities. A resource plot shows the remaining resources over time.

The various graphical items in the schedule display section may be selected using the mouse. This graphical point-and-click technique simplifies scheduling and rescheduling, allowing functions to be performed efficiently with a low risk of error.

The bottom display section provides detailed information to support different operator actions. The operator can view 3 out of 12 bottom displays at a time. Some bottom displays provide feedback to the operator during manual scheduling, while other displays show summary statistics on resource utilization and the number of scheduled and unscheduled activities.

General-Purpose Features

The concepts of generalization and abstraction were used to design and implement ROSE. By generalization, we mean expanding an application-specific requirement to

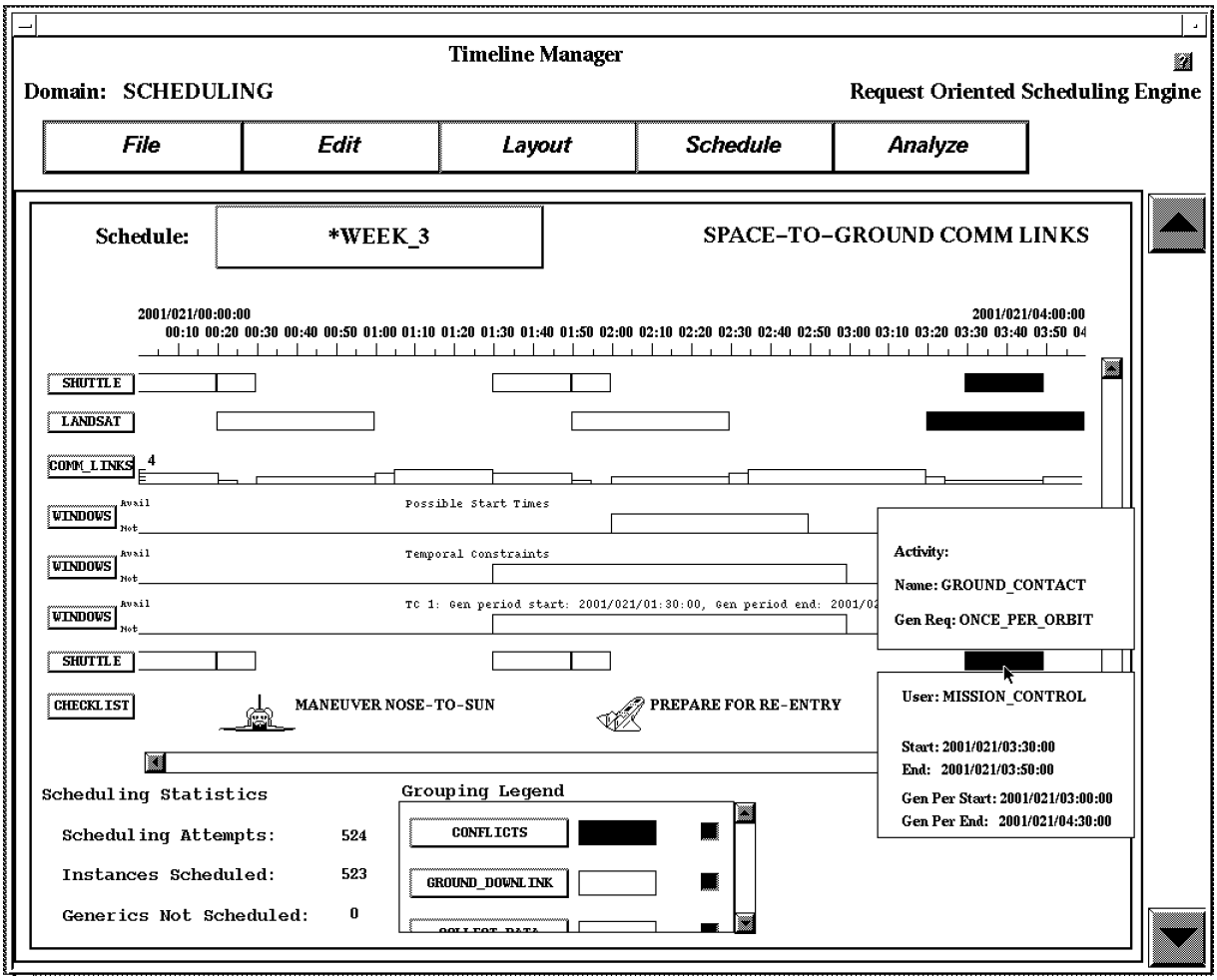

Figure 1. The ROSE Timeline Manager Screen

cover a known class of applications. For example, many NASA applications have constraints, such as:

Activity X must be scheduled
within 3 hours after
Activity Y

This requirement was extended to the general case:

Activity X must [start *or* end]
[more than *or* less than *or* exactly]
< time duration >
[before *or* after] the
[start *or* end] of
Activity Y

By abstraction, we mean taking a requirement that applies to one object in ROSE and applying it to others. For example, cut and paste functions can be applied to activities. Abstraction extends the design and application of the cut and paste functions to other items, such as timeline plots and scrolling lists.

Unconstrained use of these principles can result in increased software development cost and time. However, appropriate application of these principles can lead to lower life cycle costs. Our software maintenance activities tend to focus on implementing new capabilities rather than implementing (often tedious and enthusiasm-draining) extensions to existing capabilities.

The user interface is one feature for which we have noted a particular lack of standardization and a desire for custom development. Automated scheduling systems are often used to replace an existing manual system. Users typically want an interface that exactly duplicates the paper forms used previously.

Table 2 lists some of the general-purpose features in ROSE. For more information on ROSE, see [Weinstein, 1993].

FLEXIBILITY

Several factors contribute to ROSE's flexibility. First, the scheduler uses the Flexible Envelope Request Notation (FERN) scheduling language to specify resources and event requests. The user can specify both application-unique data and scheduling rules in FERN, thus reducing the need to change code in ROSE. Second, ROSE was designed to avoid scheduling problem size limitations. Third, the user interface can be configured to meet the needs of various scheduling domains. Finally, ROSE supports multiple operational modes for scheduling. Each of these methods is discussed separately in the following paragraphs.

The principal component of ROSE's data-driven capability is the FERN language. FERN provides a robust request specification interface to the ROSE scheduler, enabling application-unique knowledge encoding. FERN defines the interface between the users and the software scheduling system. It specifies schedule requests, resources, scheduling goals, timing constraints, and flexible start times and durations. The FERN files created to model the scheduling application are input to ROSE.

Figure 2 shows the input data types and relationships that can be represented in FERN. A generic request is used to specify repetitive events (e.g. an instrument calibration is performed once per day). Generic requests contain one or more activities to be scheduled. Activities specify a sequence of steps to be scheduled, each of which has a duration and requires resources. Steps and activities are restricted by constraints. For example, "commanding from the control center must occur only between Acquisition-of-Signal (AOS) and Loss-of-Signal (LOS)," is a constraint that restricts the time window for sending commands. Timegraphs are used to represent windows of time. For example, the timegraph AOS_LOS might contain the list of AOS and LOS times, and be used to represent all time windows for commanding during the week. Goals measure the quality of a schedule. For example, a goal might specify that the scheduler should maximize the use of a particular resource, or should schedule as many activities as possible from a particular user. ROSE will calculate the goal statistic and report the results in a statistics display. Annotations allow comments and other notes to be placed on the schedule.

The scheduling algorithms, memory management, and data structures in ROSE are designed not to limit the size of the problem that can be handled. Linked lists are used so that data structures are dynamically allocated in main memory, and sizes are limited only by the amount of available virtual memory. For example, ROSE supports any number of contingency schedules, resources, activities, steps, graphical plots, and schedule segments.

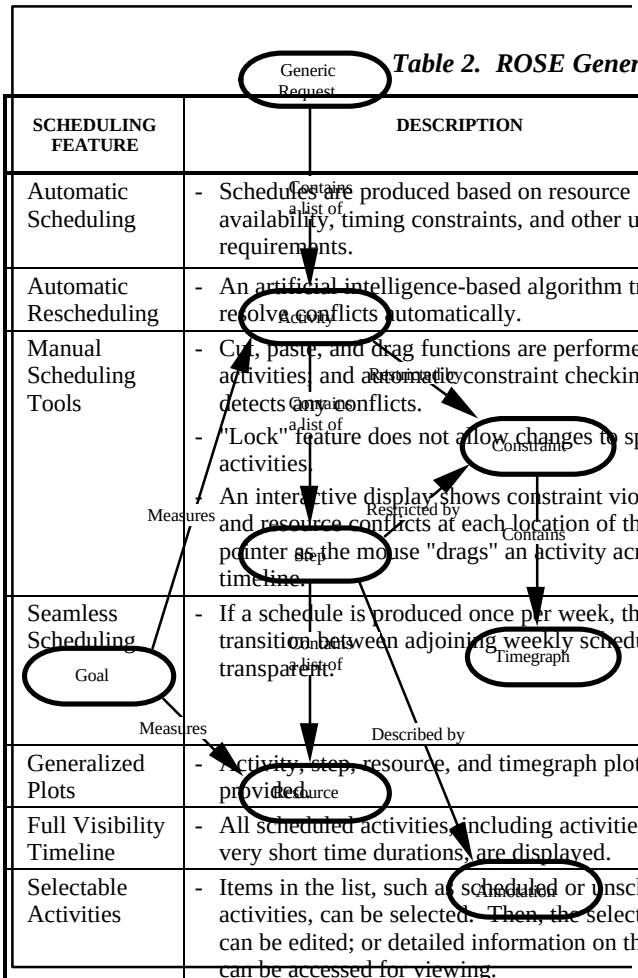

Table 2. ROSE General-Purpose Features

SCHEDULING FEATURE	DESCRIPTION	BENEFITS
Automatic Scheduling	- Schedules are produced based on resource availability, timing constraints, and other user requirements.	- The computer produces schedules quickly. - The computer checks for constraint violations and resource conflicts.
Automatic Rescheduling	- An artificial intelligence-based algorithm tries to resolve conflicts automatically.	- The operator has an additional tool to resolve conflicts quickly.
Manual Scheduling Tools	- Cut, paste, and drag functions are performed on activities, and automatic constraint checking detects any conflicts. - "Lock" feature does not allow changes to specified activities. - An interactive display shows constraint violations and resource conflicts at each location of the mouse pointer as the mouse "drags" an activity across the timeline.	- The operator controls the scheduling process. - The automatic scheduler cannot move locked activities, thus preventing inadvertent schedule changes. - The tool identifies the resources in conflict at particular times.
Seamless Scheduling	- If a schedule is produced once per week, the transition between adjoining weekly schedules is transparent.	- Activities crossing the schedule boundary are automatically carried over to the next week. - When the operator scrolls to the previous or next week, the available scheduling information is automatically loaded.
Generalized Plots	- Activity, step, resource, and timegraph plots are provided.	- The operator can view detailed data graphically using different types of plots.
Full Visibility Timeline	- All scheduled activities, including activities with very short time durations, are displayed.	- The operator can view all activities using a large time scale, such as 1 week (zoom out).
Selectable Activities	- Items in the list, such as scheduled or unscheduled activities, can be selected. Then, the selected item can be edited; or detailed information on that item can be accessed for viewing.	- A point-and-click technique accesses detailed information quickly. - This feature inserts, removes, or edits activities easily.
Marking	- Activities are assigned colors based on characteristics such as: if in conflict, resource used, or ground station used.	- Operators can assign desired colors to events to make it easier to understand the schedule.
Interruptible Scheduling Process	- The operator can watch the schedule being created and can interrupt the scheduling algorithm at any time.	- The operator can make minor adjustments to the schedule before the automatic scheduling algorithm schedules low-priority activities.
"What -If" Scheduling	- The operator can try different conflict resolution strategies and rules, creating and saving many schedules before selecting the best.	- Contingency schedules can be developed. - The best schedule is selected from competing strategies, heuristics, and algorithms.
Temporal Constraints	- This mechanism restricts the times when activities can be scheduled.	- Timing constraints are enforced (e.g., all playback and commanding activities occur between AOS and LOS).
Checkpoint/ Restart	- The scheduling process can be interrupted, and the partial schedule can be saved and restored.	- Schedules can be built incrementally, saved, and restored.

Figure 2. FERN Data Types and Relationships

The Timeline Manager screen can be tailored by the user to meet specific scheduling needs. ROSE has a generalized plot capability. As described earlier, the data may be displayed using four types of plots: activity, step, resource, and timegraph. Also, the user may bring up different sets of plots on the schedule display section to view different aspects of the scheduling problem. For example, one set of plots may depict the schedule from the perspective of the resources used for each activity; and another set of plots may depict the schedule showing levels of resource utilization over time.

Multiple operational modes, ranging from fully automatic scheduling to fully manual scheduling, allow the operator to choose the desired level of scheduling support. The operator may begin scheduling by manually placing activities on the schedule and locking them in place. The operator may then execute the software scheduling algorithm to complete the schedule, which will take into account activities already placed on the schedule. The scheduling algorithm tries to create a conflict-free schedule containing as many of the requested activities as possible. The operator may then use one or more of the manual scheduling tools to alter the schedule created by ROSE.

Compton Gamma Ray Observatory - a Case Study in ROSE Flexibility

The Compton GRO Spacecraft was deployed from the Space Shuttle in early 1990. GRO uses NASA's Tracking and Data Relay Satellite System (TDRSS) as a communications link. The GRO Mission Operations Center (MOC) transmits commands and receives telemetry and scientific data from the GRO Spacecraft through forward and return TDRSS Multiple Access (MA) low data rate and Single Access (SA) high data rate communications channels (see Figure 3). To use TDRSS services, the GRO MOC generates TDRSS service requests that are electronically transmitted to the NCC in Greenbelt, MD. NASA's Flight Dynamics Facility (FDF) provides GRO and other TDRSS customers with User Antenna View (UAV) information. This UAV information contains a list of time windows that show when the GRO Spacecraft can see (line-of-sight) one of the three geostationary TDRSS satellites.

Currently, GRO personnel visually examine UAV data to schedule requests for TDRSS services which can use any one of three TDRSS satellites. This time-consuming process can take over 4 hours per day.

In a recent prototyping effort, we successfully reutilized ROSE to provide an automated scheduling tool supporting the GRO MOC. On a Monday morning, we met with GRO personnel to discuss their scheduling needs and to obtain a sample week of FDF UAV data. During the week, we built a translator to convert the UAV data into the equivalent FERN format (a timegraph), then encoded a model of the GRO problem in FERN, and generated a custom report in the format used by GRO. On Friday of that week, we provided a demonstration that met most of their scheduling needs. In 30 seconds, ROSE automatically created a 7-day GRO contact schedule. Analysis by GRO personnel indicated that ROSE created a viable schedule.

The FERN scheduling language provided the flexibility to represent the three major types of data used by GRO. First, UAV data is converted into FERN timegraphs (see Figure 4 for a list of potential contact times when GRO can see a TDRS, and Figure 5 for the equivalent FERN timegraph).

Second, TDRSS communication channels are represented as resources with specific amounts. In this application, however, the resources need a special mask to show resource availability since GRO can see only a particular TDRS at specific time windows. The TDRSS communication channel plots (resource) intersect with UAV data (timegraph) to create a new FERN resource plot. Thus, FERN allows us to derive new timegraphs and resources from previously defined structures.

Third, FERN generic requests were created specifying the frequency with which to schedule GRO events, the time windows in which GRO events should be scheduled, and the resource requirements. The GRO scheduling heuristics were encoded into FERN without changing the ROSE code. Scheduling heuristics indicated that GRO first schedules all TDRS East MA events, followed by TDRS West MA, TDRS Zone, and finally, TDRS SA events during open UAV contacts. An additional GRO scheduling requirement stated that all events must be separated by a minimum of 2 minutes to allow for setup.

The GRO demonstration highlighted ROSE's ability to produce schedule reports used to request TDRSS services from NASA's NCC. FERN’s flexibility allowed us to provide a solution quickly without modifying the code. We reutilized the ROSE software by designing the scheduling kernel to be loosely coupled with the application-unique information represented in FERN.

EXTENSIBILITY

A layered approach promotes ROSE extensibility. A typical ROSE application consists of the ROSE Kernel Subsystem, the ROSE User Interface Subsystem, and a layer of application-unique software (see Figure 6). The ROSE Kernel and User Interface Subsystems were designed using Ada generic code templates extensively. These generic code templates make it easy to add new user interfaces, scheduling algorithms, and scheduling heuristics to the ROSE Kernel Subsystem. Replacing the scheduling algorithms and heuristics is a matter of

using the data structures, functions, and procedures which define the algorithms and heuristics, and recompiling and linking the code with the ROSE Kernel Subsystem.

Several ROSE scheduling algorithms have been implemented, and we plan to develop and implement additional algorithms. Algorithms may be written in Ada or other programming languages such as C, C++, or FORTRAN. ROSE's principal algorithm is called Earliest Possible and is designed to create a conflict-free schedule by placing each request at its earliest possible start time.

New ROSE scheduling algorithms can be implemented using the basic generic scheduling template and the four basic scheduling components shown in Figure 7. The Figure 7 components with the thick shaded border are Ada generic components used to instantiate a new scheduling algorithm.

ROSE Kernel Subsystem services include: allocating and de-allocating resources, finding event start times and

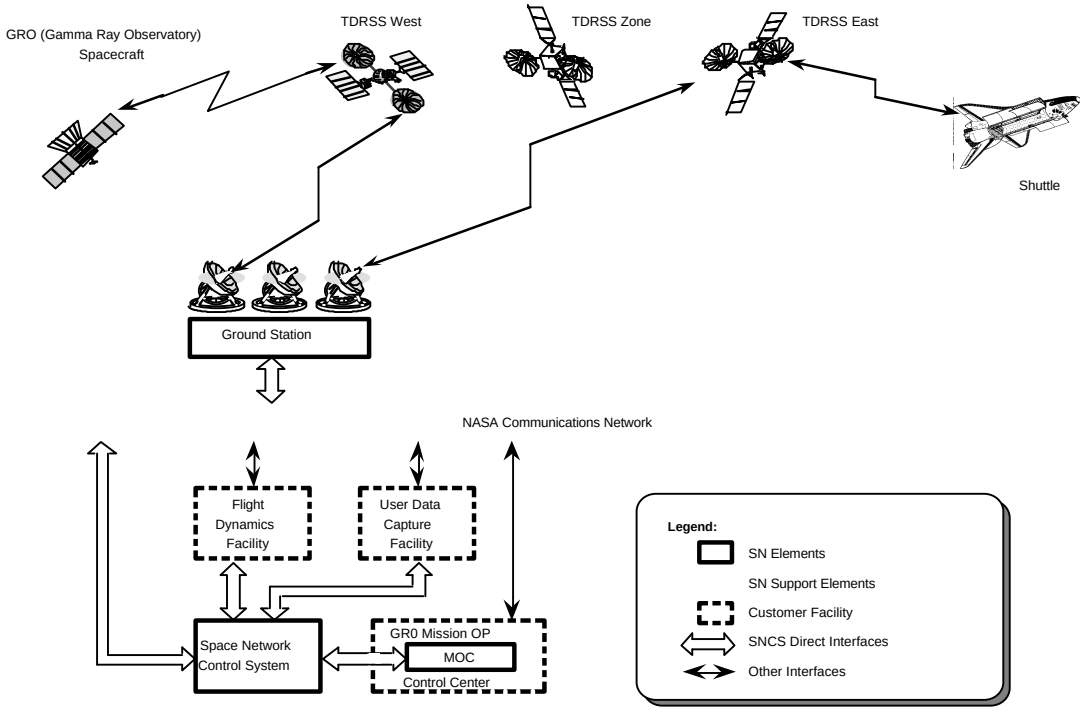

Figure 3. The TDRSS Communications Network Used by GRO

durations, detecting event and resource conflicts, finding events to schedule, adding events to the schedule, removing events, processing constraints, and FERN parsing. These services are available to the application layer for creating and modifying schedules.

Using ASCII representations for external data contributes greatly to ROSE's extensibility. Data translation utilities can be easily written to translate scheduling information from one application to another. In the ROSE application for GRO, a Practical Extraction and Report Language (PERL) script extracted scheduling information from an electronic version of a printed report and then re-formatted the data to produce FERN files.

Scheduler for Operational Resources Through an Interactive Module - a Case Study in ROSE Extensibility

The Scheduler for Operational Resources Through an Interactive Module (SORTIM) family of schedulers was developed to automate flight and academic training scheduling for Air Force pilots. It serves as an example of extending an existing software system to meet the requirements of an entirely different application or domain. The SORTIM systems were

prototypes developed for the USAF to automate scheduling activities for training student pilots. Resources included aircraft, flight simulators, students, instructors, and classrooms. ROSE reuse provided to the SORTIM development a substantial cost savings and a leap in technology. The SORTIM/ROSE schedulers provided advanced functionality that was not available in previous training schedulers.

Three SORTIM schedulers were developed using ROSE. Two schedulers required new scheduling algorithms and custom user interfaces. The third scheduler required only minimal application-layer changes. The first system automatically allocated jets and simulators for T-37 Specialized Undergraduate Pilot Training (SUPT). The second system addressed planning, scheduling, and training management needs for the KC-135 Pilot/ Navigator/Boom academic, flight, and simulator training programs.

Application-unique scheduling decisions were coded into the scheduling framework shown in Figure 7. In particular, functions were supplied to "Pick Next Event", "Pick Start Time," and "Try to Remove

TDRS/GRO
LOW GAIN ANTENNA
ATTITUDE DEPENDENT
POTENTIAL CONTACT TIMES

PREDICTION START TIME 940725.000000
PREDICTION STOP TIME 940725.150000

TDRS ANTENNA TYPE - SINGLE ACCESS

GRO INITIAL ATTITUDE (GCI J2000)

QUATERNIONS 0.6430506 0.0454778 0.7405362 0.1897983

ACADS AXES	X-AXIS	Y-AXIS	Z-AXIS
RIGHT ASCENSION	106.55	256.45	349.67
DECLINATION	69.25	18.15	9.72

INDEX NUM	CONTACT START YYMMDD.HHMMSS	CONTACT STOP YYMMDD.HHMMSS	DURAT (MIN)	TDRS ID	LGA ID	MAXIMUM ELEVATION ANGLE
1	940725.000000	940725.002200	22.00	TD-5	2	35.88
2	940725.010400	940725.020100	57.00	TD-5	2	36.70
3	940725.024300	940725.034000	57.00	TD-5	2	46.17
4	940725.042300	940725.052000	57.00	TD-5	2	63.37
5	940725.060300	940725.070100	58.00	TD-5	2	83.55
6	940725.074400	940725.084100	57.00	TD-5	1	101.35
7	940725.092400	940725.102100	57.00	TD-5	1	78.74
8	940725.110400	940725.120100	57.00	TD-5	2	105.04
9	940725.124400	940725.132200	38.00	TD-5	1	58.85
10	940725.142300	940725.144400	21.00	TD-5	1	43.16

Figure 4. Tabular Data Representing TDRSS Contacts with the GRO Spacecraft

```
-----------------------------------------------
--| FERN Timegraph Data Structure representing
--| FDF provided UAV Data for Calendar Mark
--|       DAY 206, MONDAY 07-25-1994
-----------------------------------------------
Timegraph GRO_TDRSS_WEST_UAV_CONTACTS  is
  (Create starting from Calendar Mark
   until 1994/7/25/00:00 off, until 1994/7/25/00:22 on,  --Contact#1  22 min
   until 1994/7/25/01:04 off, until 1994/7/25/02:01 on,  --Contact#2  57 min
   until 1994/7/25/02:43 off, until 1994/7/25/03:40 on,  --Contact#3  57 min
   until 1994/7/25/04:23 off, until 1994/7/25/05:20 on,  --Contact#4  57 min
   until 1994/7/25/06:03 off, until 1994/7/25/07:01 on,  --Contact#5  58 min
   until 1994/7/25/07:44 off, until 1994/7/25/08:41 on,  --Contact#6  57 min
   until 1994/7/25/09:24 off, until 1994/7/25/10:21 on,  --Contact#7  57 min
   until 1994/7/25/11:04 off, until 1994/7/25/12:01 on,  --Contact#8  57 min
   until 1994/7/25/12:44 off, until 1994/7/25/13:22 on,  --Contact#9  38 min
   until 1994/7/25/14:23 off, until 1994/7/25/14:44 on,  --Contact#10 21 min
   until forever off)
End Timegraph

-----------------------------------------------------
--| TDRSS WEST MAR   FERN Resource Data Structure
-----------------------------------------------------

Resource TDRSS_WEST_MAR is
  (Forever 1) with min gap 2 minutes
--| Flight Operations Heuristic 2 minute
--| minimum event separation mandatory
End Resource

-----------------------------------------
--| TDRSS WEST MAR GRO AVAILABILITY
--| FERN Linked Resource Data Structure
-----------------------------------------

Resource TDRSS_WEST_MAR_GRO_AVAILABILITY is
  (Link GRO_TDRSS_WEST_UAV_CONTACTS and TDRSS_WEST_MAR )
```

```
End Resource
```

Figure 5. FERN Data Generated from Figure 4 Data

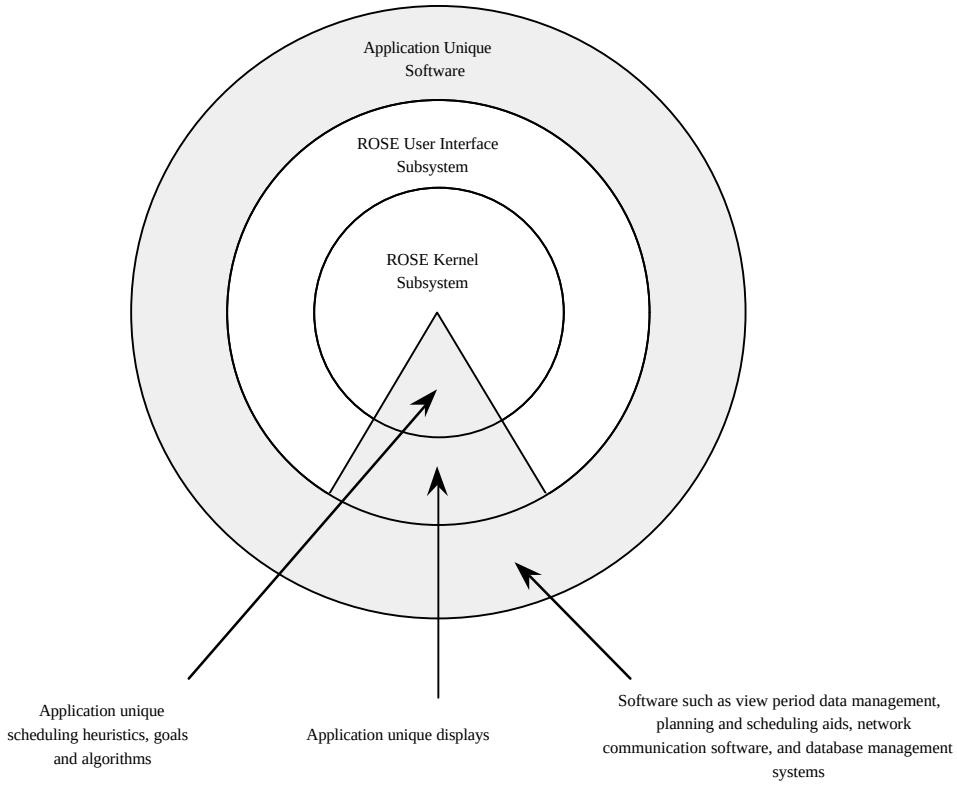

Figure 6. ROSE Layered Architecture

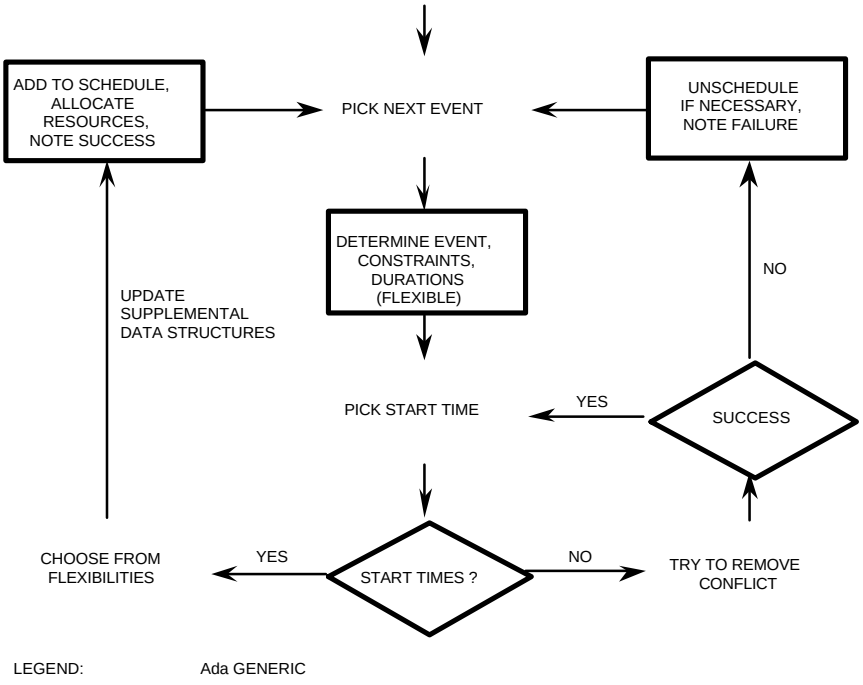

Figure 7. ROSE General Scheduling Framework

Conflict" in the generic scheduling templates. These changes were made without modifying the ROSE Kernel Subsystem (see Figure 6). The FERN language was used to represent basic scheduling rules, schedule requests, activities, steps, resources, and temporal constraints. Also, the Transportable Applications Environment Plus (TAE+) Graphical User Interface (GUI) Builder/Code Generator was used to create the new GUI, which contained buttons, pull-down menus, and selection lists.

ROSE reutilization in this entirely different application demonstrated that the "software building blocks" must be re-configurable. In this case study, the existing GUI was discarded, and a custom user interface was built to look like existing forms. The ROSE layered architecture simplified this process. Additional code implemented some scheduling rules and resources that could not be represented in FERN and incorporated application-unique scheduling decisions.

PORTABILITY

The programming language is an important factor in facilitating software portability across different workstations and computers. ROSE is written in Ada (ANSI/MIL-STD-1815A-1983). Ada has a single language reference manual, and all Ada compilers must be validated to ensure conformance to the standard semantics of the language. However, the standard Ada environment does not extend to operating system interfaces, so we followed the POSIX standard using Ada bindings. The ROSE user interaction is managed by the TAE+ User Interface Management System (UIMS), which uses X and Motif. TAE+ generates Ada code and has Ada bindings in its libraries. We have compiled the ROSE Kernel Subsystem on a number of different hardware platforms and operating systems, including: Sun 3, Sun SPARC (SunOS 4.1 and Solaris 2.3), HP PA-RISC, DEC VAX/VMS, and DEC Alpha (OSF/1 and OpenVMS).

Writing portable code can be done without implementing arbitrary limits, such as fixed sizes for arrays and fixed fields for input data. Besides supporting traditional data structures, such as trees and linked lists, Ada provides flexible constructs, including variant records, discriminant records, unconstrained arrays, and pointers (access types), which remove limits on problem type and size without restricting portability. Generic packages and generic procedures promote both portability and reusability, especially for general purpose data handling routines, such as linked lists. As discussed previously, the heart of the scheduling kernel is a generic procedure. The user can create many different schedules by trying different scheduling algorithms and heuristics.

All external data formats used by ROSE have an ASCII representation, which facilitates portability, debugging, and extensibility. Portability is enhanced because different hardware platforms, particularly those with different binary representations, can still exchange scheduling information. In one case, we augmented the ASCII format with a binary format to improve performance. However, if the binary format file does not exist, ROSE automatically uses the ASCII format file. Thus, both performance and portability are maintained. Debugging is enhanced because a user can read and understand the external data residing on disk.

Porting from VMS to UNIX - a Case Study in ROSE Portability

In 1991, ROSE was ported from a VMS workstation to a UNIX workstation. We had developed most of ROSE on a VAX 11/780, and then later on VAXstations, using VMS. For the port to UNIX, most code changes resulted from the foreign language bindings (POSIX, X, and TAE+), because not all of the interface pragmas were part of the Ada standard. Also, the bindings themselves had to be changed. Other changes were required to accommodate differences in file name and directory structures on different platforms.

Presently, we ensure portability with a common source tree (except for foreign language bindings) for VMS and SunOS. Ada does not define a preprocessor for conditional compilation, so all code must work for both platforms. To maintain portability, all new code changes must compile cleanly on both platforms, and both executables must pass all regression tests. In general, all ROSE ports to various platforms have gone smoothly.

FUTURE DIRECTION FOR ROSE

ROSE has been proposed as the core scheduling component for the upcoming replacement of the Service Planning Segment (SPS) of the NCC at Goddard Space Flight Center (GSFC). The SPS is the software suite responsible for scheduling and coordinating TDRSS support. System planners at GSFC are currently evaluating candidate software and hardware components for the SPS replacement.

Because of its adherence to open systems standards, we are confident ROSE will run on the chosen open-system architecture. ROSE provides the flexibility to support the dynamic requirements of the SPS. The FERN language is well suited for use in an environment where scheduling needs change frequently. Additionally, ROSE's dynamic allocation (using linked lists) accommodates growth. The SPS replacement requires specialized scheduling algorithms and heuristics, as well as a custom user interface. Using an extensible design facilitates the addition of new code into ROSE.

CONCLUSION

ROSE has been demonstrated to be reusable for solving a broad range of scheduling problems, including spacecraft, TDRSS, and pilot training scheduling. A layered, reusable design has resulted in a system that can be tailored to schedule a large number of event requests efficiently in a variety of applications. ROSE reuse reduces the time to develop a custom scheduling system.

CITED REFERENCES

[Brouchard, 1992] Brouchard, *Intelligent Training Management System Scheduler (ITMSS)*, Briefing Material, Miami Valley Research Institute, Dayton, OH (June 1992).

[Moe, et al., 1994] Karen Moe, *Space Network Flexible Scheduling Enhancements*, White Paper, Goddard Space Flight Center, Greenbelt, MD (March 7, 1994).

[Weinstein, 1993] Stuart Weinstein, *Request-Oriented Scheduling Engine User's Guide*, Goddard Space Flight Center, Greenbelt, MD (May 1993).